Transfer Learning via Multiple Inter-Task Mappings

Anestis Fachantidis[1], Ioannis Partalas[1], Matthew E. Taylor[2], and Ioannis Vlahavas[1]

[1] Department of Informatics, Aristotle University of Thessaloniki
{afa,partalas,vlahavas}@csd.auth.gr
[2] Department of Computer Science, Lafayette College
taylorm@cs.lafayette.edu

Abstract. In this paper we investigate using multiple mappings for transfer learning in reinforcement learning tasks. We propose two different transfer learning algorithms that are able to manipulate multiple inter-task mappings for both model-learning and model-free reinforcement learning algorithms. Both algorithms incorporate mechanisms to select the appropriate mappings, helping to avoid the phenomenon of negative transfer. The proposed algorithms are evaluated in the Mountain Car and Keepaway domains. Experimental results show that the use of multiple inter-task mappings can significantly boost the performance of transfer learning methodologies, relative to using a single mapping or learning without transfer.

1 Introduction

In recent years, a wealth of *transfer learning* (TL) methods have been developed in the context of *reinforcement learning* (RL) tasks. Typically, when an RL agent leverages TL, it uses knowledge acquired in one or more (*source*) tasks to speed up its learning in a more complex (*target*) task.

Although the majority of the work in this field presumes that the source task is connected in an obvious or natural way with the target task, this may not the case in many real life applications where RL transfer could be used. These tasks may have different state and action spaces, or even different reward and transition functions. One way to tackle this problem is to use functions that map the state and action variables of the source task to state and action variables of the target task. These functions are called *inter-task mappings* [12].

While inter-task mappings have indeed been used with successfully in several settings, we identify several shortcomings. First, an agent typically uses a hand-coded mapping, requiring the knowledge of a domain expert. If human intuition cannot be applied to the problem, selecting an inter-task mapping may be done randomly, requiring extensive experimentation and time not typically available in complex domains or in real applications. On the other hand, even if a correct mapping is used, it is fixed and applied to the entire state-action space, ignoring the important possibility that different mappings may be better for different regions of the target task.

This paper examines the potential impact of using multiple mappings in TL. More specifically, we propose two different transfer learning algorithms that are able to manipulate multiple inter-task mappings for both model-learning and model-free reinforcement learning algorithms. Both algorithms incorporate mechanisms for the selection of the appropriate mappings in order to avoid the *negative transfer* phenomenon, in which TL can actually decrease learning performance. The proposed algorithms are evaluated in the domains of Mountain Car and Keepaway and experimental results show that using multiple inter-task mappings can improve the performance of transfer learning.

2 Transfer via Multiple Inter-Task Mappings

In order to enable a transfer learning procedure across tasks that have different state variables and action sets, one must define how these tasks are related to each other. One way to represent this relation is to use a pair (X_S, X_A) of inter-task mappings [12, 8], where the function X_S maps a target state variable to a source state variable and X_A maps an action in the target task to an action in the source task.

Inter-task mappings have been used for transferring knowledge in several settings like TD-learning [12], policy search and model-based algorithms [13, 9]. All these approaches use only one pair of inter-task mappings which is usually dictated by a human expert.

Defining multiple mappings of the target task to the source task is a domain-dependent process and requires the involvement of a domain expert. Methods that automatically construct the mapping functions have been recently proposed [10], but are currently very computationally expensive and outside the scope of this paper. Additionally, one could generate *all* feasible mappings and use them. A problem of this approach is that in domains with many features, generating the mappings becomes prohibitive in terms of both computational and memory requirements.

2.1 Transferring with Multiple Inter-Task Mappings in Model Based Learners

Model-based Reinforcement Learning algorithms, such as Fitted R-max [1] and PEGASUS [3], use their experiences to learn a model of the transition and reward functions of a task. These models are used to either produce new mental experiences for direct learning (e.g., Q-Learning) or with dynamic programming methods and planning in order to construct a policy. Contrary to direct learning agents, model based RL agents have some added options for transfer learning as they can also transfer the transition and/or reward function of a source task. Models of these functions can have many representations, such as a batch of observed instances or a neural network that approximates its transition and/or reward function.

Transferring instances in model-based RL has been proposed in the TIMBREL algorithm [9], in which a single-mapping TL algorithm showed significant improvement over the no-transfer case. Our proposed method builds upon the TIMBREL algorithm, but can autonomously use multiple inter-task mappings.

In addition to empirically demonstrating the effectiveness of using an instance-based RL algorithm with multiple mappings, we will also argue the existence of a theoretical connection between the described setting and another setting more extensively studied: multi-task, single-mapping transfer. In a multi-task transfer learning problem, an agent is transferring knowledge from more than one source task into a target task that is more complex, but still related to the source tasks. In order to avoid negative transfer, the agent must decide what information to transfer from which source task, and when to transfer this information [2].

In this case, recently proposed methods can assist the agent when selecting the appropriate source task, as well as when and what knowledge to transfer. Lazaric et al. [2] defines two probabilistic measures, *compliance* and *relevance*. These measures can assist an agent to determine the most similar source tasks to the target task (compliance) and also to select which instances to transfer from that source task (relevance). Results showed that this is a feasible and efficient method for multi-task trasfer—we will extend this method to use multiple inter-task mappings, rather than multiple source tasks.

We consider each inter-task mapping function as a hypothesis, proposed to match the geometry and dynamics between the source and target task. By considering a source task through this hypotheses, we directly differentiate the source task's outcomes for a fixed input. Mapping states and actions from a target task to a source task not only transforms the way we view and use the source task, but also the way it behaves and responds to a fixed target task's state-action query (before these are mapped). Thus, every mapping X_i can be considered as a constructor of a new *virtual source task* S_{X_i}. This naturally **re-formulates the problem of finding the best mapping as a problem of finding the most compliant virtual source task**. Additionally, this re-formulation transforms the problem of finding out when to sample from a certain mapping to a problem of sample relevance.

Based on previously defined notation [2], we define the compliance of a target task transition τ_i and a mapping X_i as:[3]

$$\lambda_{X_i} = \frac{1}{Z^P Z^R}(\sum_{j=1}^{m} \lambda_{ij}^P)(\sum_{j=1}^{m} \lambda_{ij}^R)$$

[3] P and R refer respectively to the transition and reward functions of the target task. Z^P and Z^R are normalization constants detailed elsewhere [2].

The compliance between the target task (not only one instance of it) and the virtual source task S_{X_i} generated by the mapping X_i is defined as:

$$\Lambda = \frac{1}{t} \sum_{i=1}^{t} \lambda_i P(s)$$

To implement this new definition of compliance we design a novel multiple-mappings TL algorithm, *COmpliance aware transfer for Model Based REinforcement Learning* (COMBREL), which is able to select mappings based on their compliance as defined above. The equally important notion of relevance is left for future work.

First, COMBREL implements a simple and straightforward segmentation of the state-action space, which we call *mapping regions.* These regions allow for different compliance values—and thus different best mappings—for different regions of the state-action space. This adds the flexibility that each state-action region has its own best mapping instead of computing one best mapping for the whole target task. Additionally, every mapping region can be considered as a different target task (just like each mapping may be considered a different source task). A compliance value is calculated for each mapping region, averaging the compliance of every state-action instance in the region. It is important to note that mapping regions are created based on a fixed resolution. In this paper, a low resolution of four mapping regions, per state variable, was selected assisting on the efficiency of the method and on its actual usefulness since, as (intuitively) the higher the resolution, the less the information we can obtain for each new (thinner) segment of the state space. Its important to note that setting the resolution of the mapping region's segmentation is, for the time being, an experimental choice and not an informed decision.

For COMBREL's underlying model-based algorithm we use Fitted R-Max, building upon the single-mapping transfer algorithm TIMBREL [9]. TIMBREL cooperates with a model-based RL algorithm like Fitted R-Max by assisting its model approximation, by transferring source task instances near the points that need to be approximated. In the absence of such source task instances, no transfer takes place. However, the specific use of Fitted R-Max should not be considered as a limitation of the method as COMBREL is easily adaptable to other instance-based RL algorithms.

The flow of our proposed method and algorithm, COMBREL (see Algorithm 1), is as follows. On lines 1-5, the algorithm records source task transitions and then starts training in the target task. A set of regions is defined, segmenting the state-action space so each can have a different best mapping. Lines 5-8 compute the compliance of the last target task transition with each of the mappings, X_i. This value is then added to the regional compliance estimate. On lines 9 and 10, if the agent's model-based algorithm is unable to approximate a state with its current data, it starts transferring source task data. To select the best mapping for the current agent's state, on lines 11 to 15 the agent decides on the average compliance of the current region for each mapping (i.e., virtual source task).

Algorithm 1 COMBREL - Multiple Mappings in Model Based Learners

```
1: Learn in the source task, recording $m$ $(s, a, r, s')$ transitions.
2: while training in the target task do
3:     for ($e$ epidodes) do
4:         Define $D_i$ regions of the state-action space in the target task
5:         Record Target Task transitions $\langle x_T, a_T \rangle$
6:         Find Region membership $D$ for each $\langle x_T, a_T \rangle$
7:         For each Mapping $X_i$ define a virtual source task $S_i$
8:         Update region's compliance estimate, adding compliance $\lambda_i$ of $\langle x_T, a_T \rangle$ with
           each virtual source task $S_i$
9:     if the model-based RL algorithm is unable to accurately estimate $T_T(x_T, a_T)$
       or $R_T(x_T, a_T)$ then
10:        while $T_T(x_T, a_T)$ or $R_T(x_T, a_T)$ does not have sufficient data do    ▷ Use
           the most compliant mapping to save instances near $\langle x_T, a_T \rangle$
11:            Find $|X|$ pairs $(X_S, X_A)$ of inter-task mappings
12:            Find Region membership of $\langle x_T, a_T \rangle$
13:            for ($|X|$ iterations) do
14:                Find $X^{best}$ based on current region's compl. estimate $\Lambda_{D_i}$ with $X_i$
15:            $X^{best} \leftarrow X_{\max(\Lambda_{D_i})}$                    ▷ The most compliant
16:            Translate samples from $S^{best}$ and put them in $\hat{T}$
17:            Locate 1 or more instances in $\hat{T}$ near the $\langle x_T, a_T \rangle$ to be estimated.
```

When found, it translates samples from it and combines them with target task samples to create a model. Then, the algorithm iterates and continues learning.

2.2 Multiple Inter-Task Mappings in TD Learners

Having discussed using multiple inter-task mappings in model-based RL methods, we now turn our attention to model-free, temporal difference agents.

We assume that an RL agent has been trained in the source task and that it has access to a function $Q_{source}(s, a')$, which returns an estimation of the Q value for a state s' and action a' of the source task. The agent is currently being trained in the target task, learning a function $Q_{target}(s, a)$ and senses the state s. In order to transfer the knowledge from the source task, we find the best mapping and we add the corresponding values from the source task via the selected mapping. Algorithm 2 shows the pseudocode of the transferring procedure.

On lines 3–10, the algorithm finds the best mapping from instances that are recognized in the target task. More specifically, we define the best mapping as the one with the maximum mean sum of the values for each action in the source task (lines 6–8). There are alternate ways to select the best mapping, such as finding the maximum action value, but we leave such explorations to future work.

After the best mapping is found, the algorithm adds to the Q-values from the source task to the Q-values from the target task (lines 11-12). We use a mapping function, g_A, which is an inverse function of f_A and maps a source action to its

Algorithm 2 Value-Addition procedure: Multiple Mappings in TD Learners

```
1: procedure VALUE-ADDITION($s, M_S, M_A, Q_{target}, Q_{source}$)
2:     $bestMeanQValue \leftarrow 0$; $bestMapping \leftarrow 0$
3:     for $i \leftarrow 1 \dots N$ do
4:         $s' \leftarrow M_S^i(s)$; $meanQValue \leftarrow 0$
5:         for all $a' \in A_{source}(s')$ do
6:             $meanQValue \leftarrow meanQValue + Q_{source}(s', a')$
7:         $meanQValue \leftarrow meanQValue / |A_{source}(s')|$
8:         if $meanQValue > bestMeanQValue$ then
9:             $bestMeanQValue \leftarrow meanQValue$; $bestMapping \leftarrow i$
10:    $s' \leftarrow M_S^{bestMapping}(s)$
11:    for $a' \in A_{source}(s')$ do
12:        $Q_{target}(s, g_A^{bestMapping}(a')) \leftarrow Q_{target}(s, g_A^{bestMapping}(a')) + Q_{source}(s', a')$
```

equivalent target action. Note that if a target action is not mapped to a source action, the algorithm does not add an extra value. Depending on the impact of the source task's Q-values, the agent will now be biased to select actions from the set $A_a = \{g_A^{bestMapping}(a')|a' \in A_{source}^{s'}\}$. The impact of the source task depends strongly on how much time was spent learning it. Specifically, if the source task was trained for a small number of episodes then the Q-values will be also small and afterwards the agent will probably select actions from A_a for a small period (assuming pessimistic initialization). As learning proceeds, the values of the target Q-function will increase and the initial bias will be overridden.

3 Domains

3.1 Mountain Car

For the mountain car domain we use the version proposed by [4]. In the standard 2D task an underpowered car must be driven up to a hill. The state of the environment is described by two continuous variables, the horizontal position $x \in [-1.2, 0.6]$, and velocity $v_x \in [-0.007, 0.007]$. The actions are {Neutral, Left and Right} move the car to the goal state, as quickly as possible. The 4D mountain car extends the 2D task by adding an extra spatial dimension [9]. The state is composed by four continuous variables (4D), the coordinates in space x, and $y \in [-1.2, 0.6]$, as well as the velocities v_x and $v_y \in [-0.07.0.07]$. The available actions are {Neutral, West, East, South, North}.

3.2 Keepaway

Keepaway [6], is a subset of the RoboCup robot soccer domain, where K keepers try to hold the ball for as long as possible, while T takers (usually $T = K-1$) try to intercept the ball. The agents are placed within a fixed region at the start of each episode, which ends when the ball leaves this region or the takers intercept it.[4]

The task becomes harder as extra keepers and takers are added to the fixed-sized field, due to the increased number of state variables on one hand, and the increased probability of ball interception in an increasingly crowded region on the other.

4 Experiments and Results

4.1 Transferring with COMBREL in Mountain Car 4D

To test the efficiency of the model-based proposed approach we conducted a series of experiments in the Mountain Car 4D domain. First, a standard Q-learning agent learned Mountain Car 2D while recording instances from its experiences. These instances, in the form of tuples $\langle s, a, r, s' \rangle$, formed a dataset of 100,000 instances, covering the state-space. The number of experiences gathered from Mountain Car 2D allowed us to create multiple datasets of varying size. These datasets will be used as the source task knowledge.

The first experiment was conducted using a fixed sample size (2,000 instances). We tested COMBREL and TIMBREL with Multiple Mappings (TIMBERL+MM) (no selection mechanism) against eight versions of single-mapping TIMBREL. Each of these versions used a different state-action mapping. This experiment revealed the importance of using multiple mappings when we have limited experimentation time and no knowledge or intuition of the best mappings that should be used. COMBREL outperformed single mapping TIMBREL for any of its versions, in their first run.

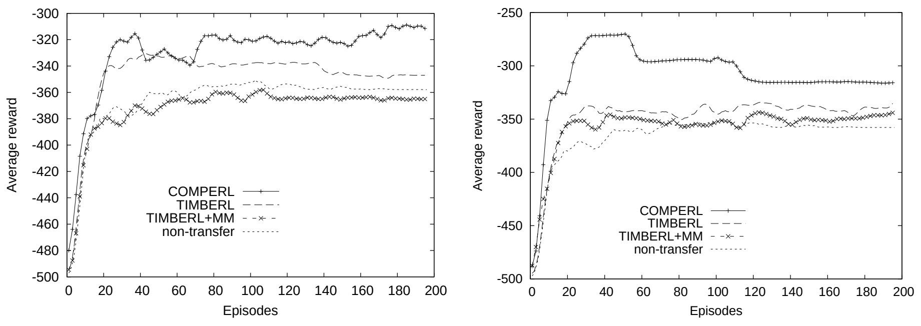
TIMBREL with its best mapping, identified in the previous experiment, is tested against the multiple mappings versions in three more experiments, using 1,000, 5,000, and 10,000 source task instances . This experiment compares the performance of COMBREL against TIMBREL and TIMBREL+MM, independently of the sample size. Furthermore this experiment allows us to analyse the sample efficiency of the three algorithms while varying the source task sample size.

About the set of parameters used for Fitted R -Max and TIMBREL, these were the same as [9]. In our experiments COMBREL, TIMBREL+MM and TIMBREL used a model breadth parameter $b = 0.4$ and a minimum fraction parameter of 10%

We ran 20 trials for each of the compared methods and for each of the source task sample sizes. Figure 1(a) shows the performance of the compared algorithms when transferring 1,000 source task instances. There is a clear performance increase with the use of COMBREL, when compared to TIMBREL+MM and the No-Transfer case (the difference was statistically significant at a 95% level). When compared to TIMBREL the gain is still statistically significant, but only for episodes 80 and above. Figure 1(b) shows results from a similar experiment,

[4] For more information please refer to the original paper [6].

but with a 2,000 instances from the source task. COMBREL demonstrated an even greater performance increase, but it was more unstable, showing the sensitivity of the multiple mappings methods with the transferred sample size. Experiments with 5,000 and 10,000 source task instances showed a similar and more stable performance gain (but are omitted for space constraints). It is important to note the poor performance of TIMBREL+MM compared to the other algorithms. This shows that using a set of mappings simultaneously, without previous domain knowledge and any selection mechanism, can result to negative transfer. Table 1 shows the four state mappings used by COMBREL in our ex-



(a) 1000 samples (b) 2000 samples

Fig. 1. Average reward in a period of 200 episodes. The curves are smoothed with a window of 10 episodes

periments, as well as the number of times they were selected. We can observe that the more "intuitively correct" mappings were found and selected most often. Mappings which seem irrelevant were also used some times, but likely in state space regions where they were useful.

4D state var.	Mapping 1	Mapping 2	Mapping 3	Mapping 4
$\mathbf{x}$	$\mathbf{x}$	$\mathbf{x}$	$\mathbf{v_x}$	$\mathbf{x}$
$\mathbf{y}$	$\mathbf{x}$	$\mathbf{v_x}$	$\mathbf{v_x}$	$\mathbf{x}$
$\mathbf{v_x}$	$\mathbf{v_x}$	$\mathbf{x}$	$\mathbf{x}$	$\mathbf{v_x}$
$\mathbf{v_y}$	$\mathbf{v_x}$	$\mathbf{v_x}$	$\mathbf{x}$	$\mathbf{v_x}$
Times used	269.992	52.075	29.966	322.411

Table 1. Four state space mappings in Mountain Car 4D and the number of times they were selected by COMBREL, in an indicative episode.

4.2 Transferring with Value-Addition in Keepaway

This section evaluates the proposed approach in Keepaway. The dimensions of the keepaway region is set to 25m $\times$ 25m and remains fixed for all source and target tasks. The algorithm that is used to train the keepers is the SMDP variation of Sarsa(λ) [7]. Additionally, we use linear tile-coding for function approximation with settings shown to work well in the Keepaway domain [5].[5]

To evaluate the performance of the proposed approach we use the *time-to-threshold* metric [12], which measures the time required to achieve a predefined performance threshold in the target task. Typically, the threshold is set empirically, after preliminary experimentation in the target task. In our case this threshold corresponds to a number of seconds that keepers maintain ball possession. In order to conclude that the keepers have learned the task successfully, the average performance of 1,000 consecutive episodes must be greater than the threshold. We compare (1) the time-to-threshold without transfer learning with (2) the time-to-threshold with transfer learning plus the training time in the source task.

Finally, an important aspect of the experiments concerns the way that the mapping will be produced. For the Keepaway domain, the production of the mappings can semi-automatic. More specifically, any K^t vs. T^t task can be mapped to a K^s vs. T^s task, where $K^s < K^t$ and $T^s < T^t$, simply by deleting $K^t - K^s$ teammates and $T^t - T^s$ opponents of the keeper with ball possession. The actual number of different mappings is:

$$\binom{K^t - 1}{K^s - 1}\binom{T^t}{T^s} = \frac{(K^t-1)!T^t!}{(K^s-1)!(K^t-K^s)!T^s!(T^t-T^s)!}$$

Transfer into 4 vs. 3 from 3 vs. 2 This subsection evaluates the performance of the proposed approach on the 4 vs. 3 target task using a threshold of 9 simulated seconds. We use the 3 vs. 2 task as the source and experiment with different number of training episodes, ranging from 0 (no transfer) to 3,200.

Table 2 shows the training time and average performance (in seconds) in the source task, as well as time-to-threshold and total time in the target task for different amount of training episodes in the source task. The results are averaged over 10 independent runs and the last column displays the standard deviation. The time-to-threshold without transfer learning is about 13.38 simulated hours.

We first notice that the proposed approach leads to lower time-to-threshold in the target task compared to the standard algorithm that does not use transfer learning. This is due to the fact that the more the training episodes in the source task the better the Q-function that is learned. Note that for 800 episodes of the 3 vs. 2 task, the keepers are able to hold the ball for an average of 8.5 seconds, while for 1600 episodes their performance increases to 12.2 seconds. As the number

[5] We use 32 tilings are used for each variable. The width of each tile is set to 3 meters for the distance variables and 10 degrees for the angle variables. We set the learning rate, α, to 0.125, ϵ to 0.01 and λ to 0. These values remain fixed for all experiments.

3 vs. 2			4 vs. 3		
#episodes	train time	performance	time-to-thr.	total time	st. dev.
0	0	-	13.38	13.38	2.02
100	0.11	4.38	13.19	13.30	1.77
200	0.23	4.67	12.59	12.82	2.10
400	0.72	6.71	12.08	12.80	1.70
800	1.73	8.52	10.28	12.01	0.97
1600	4.73	12.20	3.48	**8.21**	1.16
2500	8.42	16.02	4.16	12.44	0.60
3200	12.17	16.84	**2.76**	14.95	0.28

Table 2. Training time and average performance in the source task, as well as time-to-threshold and total time in the target task. The best time-to-threshold and total time are in **bold**.

of the training episodes in the source task increase the time that is required to reach the threshold decreases, showing that our method successfully improves performance in the target task.

The total time of the proposed approach in the target task is also less than the time-to-threshold without transfer learning in many cases. The best performance is 8.21 hours, which corresponds to a reduction of 38.6% of the time-to-threshold without transfer learning. This performance is achieved when training the agents for 1600 training episodes in the source task. This result shows that rather than directly learning on the target task, it is actually faster to learn on the source task, use our transfer method, and only then learn on the target task.

In order to detect significant difference among the performances of the algorithms we use the paired t-test with 95% confidence. We perform seven paired t-tests, one for each pair of the algorithm without transfer learning and the cases with transfer learning. The test shows that the proposed approach is significantly better when it is trained with 800 and 1600 episodes.

Scaling up to 5 vs. 4 We also test the performance of the proposed in the 5 vs. 4 target task. The 5 vs. 4 threshold is set to 8.5 seconds. The 3 vs. 2 task with 1,600, 2,500 and 3,200 training episodes and the 4 vs. 3 task with 4,000 and 6,000 training episodes are used as source tasks. Table 3 shows the training times, time-to-threshold and their sum for the different source tasks and number of episodes averaged over 10 independent runs along with the standard deviation.

Firstly, we note that in all cases the proposed approach outperforms learning without transfer learning. It is interesting to note that the best time-to-threshold is achieved when using 3 vs. 2 as a source task is achieved with fewer episodes than when using 4 vs. 3 as a source task. This means that a relatively simple source task may provide *more* benefit than a more complex source task. In addition, the 3 vs. 2 task requires less training time, as it is easier than the 4 vs. 3 task. We perform 5 paired t-tests, one for each case of the proposed approach against learning without transfer. The proposed method achieves statistically significantly higher performance (at the 95% confidence level) in all cases.

	source		5 vs. 4		
task	#episodes	tr. time	time-to-thr.	total time	st. dev.
-	0	0	26.30	26.30	2.85
3 vs. 2	1600	4.73	15.54	20.27	3.24
3 vs. 2	2500	8.42	8.88	17.31	3.23
3 vs. 2	3200	12.17	**3.43**	**15.60**	1.24
4 vs. 3	4000	7.52	10.07	17.59	1.49
4 vs. 3	6000	12.32	9.26	21.58	1.92

Table 3. Average training times (in hours) for 5 vs. 4. The results are averaged over 10 independent trials. The best time-to-threshold and total time are in **bold**.

5 Related Work

[9] proposes an algorithm that implements Transfer Learning for Model - Based RL agents. It is based on transferring instances from a source task to a target task and using them with the model-based RL algorithm Fitted R-Max to assist it in the construction of a model of the target task. Although it demonstrates promising results and a significant performance gain it uses only one hand-coded mapping between the source and the target task. Our proposed model-based multiple mappings method is based on TIMBREL but extends it with the use of a multiple mappings mechanism able to autonomously select mappings while learning.

[12] constructs an autonomous mapping selection method (MASTER) which is able to select a mapping based on the similarity between transitions in the source and target task . MASTER learns a model of the action effects in the source task. In the target task, it first selects actions randomly, sampling target task transitions. It then compares this transitions with queries on the source task model to calculates the error (difference). It selects the best mapping to minimize this error. Our model-based method is also based on the similarity of the effects between the two tasks but it makes an off-line calculation of the error thus, not spending agent time with random environment interaction. Also as our method is built up upon TIMBREL, it requires no explicit model learning in the source task as it transfer only instance from it.

In [8], each possible inter-state mapping is considered as an expert and with the use of a multi-armed bandit algorithm the agent decides which mapping to use. This method shows significant improvement but its performance is surpassed in the long run as it continues to explore always taking “advice” from low return experts.

For space limitations reasons, the reader is directed to read more about the various transfer learning methodologies in more comprehensive treatments [11].

6 Conclusions and Future Work

In this paper, we examined the benefit of using multiple-mappings in the transfer learning setting. To avoid negative transfer a multiple-mappings method must be supported by a mechanism able to select the most compliant and relevant mappings.

Results on our proposed methods showed a statistically significant benefit over the single mapping transfer algorithm as also from the multiple mapping method that uses all the mappings simultaneously. Future work includes discovering more sophisticated ways to discretize the state space in COMBREL, avoiding discontinuities in the mapping selection function between nearby states in the state space. Furthermore, whereas COMBREL implements the novel notion of a mappings compliance, this scheme can be extended to another important probabilistic measure, that of relevance [2], where we would not only care about the best mapping but also on when to sample from it and how much.

References

1. N. Jong and P. Stone. Model-based exploration in continuous state spaces. *Abstraction, Reformulation, and Approximation*, pages 258–272, 2007.
2. Alessandro Lazaric. *Knowledge Transfer in Reinforcement Learning.* PhD thesis, Politecnico di Milano, 2008.
3. A. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang. Autonomous inverted helicopter flight via reinforcement learning. *Experimental Robotics IX*, pages 363–372, 2006.
4. Satinder P. Singh and Richard S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22(1-3):123–158, 1996.
5. Peter Stone, Gregory Kuhlmann, Matthew E. Taylor, and Yaxin Liu. Keepaway soccer: From machine learning testbed to benchmark. In *RoboCup-2005: Robot Soccer World Cup IX*, pages 93–105, 2006.
6. Peter Stone and Richard Sutton. Keepaway soccer: A machine learning test bed. In *RoboCup 2001: Robot Soccer World Cup V*, pages 207–237. Springer-Verlag, 2002.
7. Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning, An Introduction.* MIT Press, 1998.
8. Erik Talvitie and Satinder Singh. An experts algorithm for transfer learning. In *Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 1065–1070, 2007.
9. Matthew E. Taylor, Nicholas K. Jong, and Peter Stone. Transferring instances for model-based reinforcement learning. In *European conference on Machine Learning and Knowledge Discovery in Databases*, pages 488–505, 2008.
10. Matthew E. Taylor, Gregory Kuhlmann, and Peter Stone. Autonomous transfer for reinforcement learning. In *7th international joint conference on Autonomous agents and multiagent systems*, pages 283–290, 2008.
11. Matthew E. Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(1):1633–1685, 2009.
12. Matthew E. Taylor, Peter Stone, and Yaxin Liu. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8:2125–2167, 2007.
13. Matthew E. Taylor, Shimon Whiteson, and Peter Stone. Transfer via inter-task mappings in policy search reinforcement learning. In *6th international joint conference on Autonomous agents and multiagent systems*, pages 37:1–37:8, 2007.